AES (Advanced Encryption Standard) is a symmetric and iterative block cipher encryption algorithm. It has a block length of 128 bits and a key length of 128, 192 or 256-bits. If the length isn’t provided, the CryptoJS library generates a 256-bit key using the passphrase given. It works by encrypting the data with substitutions and permutations. When the data is processed, it is done so as a 16 x 16-byte matrix. The algorithm consists of a variable number of rounds, each with 4 stages. If the key is 128-bits, the algorithm will run for 10 rounds. For 192-bit this increases to 12, and for 256-bit it increases again to 14. Each round has its own 128 bit “round key” that is generated from the original key. The first stage in encrypting the plaintext is to arrange the input into a 16x16-byte matrix. In the second stage, each of the rows is shifted to the left except the first row. The second row is shifted one byte, the third is shifted two bytes, and the last row is shifted three bytes. The bytes that overflow are moved into the empty space that is left where the first byte started. After this is done, stage three takes each column and uses a mathematical function to output four new bytes for each column. This new matrix is then changed back to 128 bits and XORed with the 128 bits of the round key. If it is the last round, then the resulting ciphertext is outputted, but if it isn’t the last round, then the output becomes the input for first stage of the next round. The resulting ciphertext will always be 128-bits. The decryption process is the reverse of the encryption.

DES (Data Encryption Standard) is an early Feistel Cipher encryption algorithm. It has a block size and a key size of 64-bits, but since 8 of the bits aren’t used for the key, it essentially has a key size of 56-bits. There are 16 rounds in the cipher, and each round has a 48-bit key generated from the 56-bit cipher key, which means that there are sixteen 48-bit keys used in the encryption algorithm. The first stage in the algorithm is to perform the initial permutation of the data. The second stage is to run the data through 16 rounds of the DES function, which applies a 48-bit key to the rightmost 32 bits. This is done by expanding the 32-bit block into 48 bits using an expansion permutation(duplicating half of the bits). The result of this is then combined with one of the 16 subkeys using an XOR operation. The result of the previous step is then divided into eight 6-bit pieces and each piece is processed with a “substitution box”. This is where each piece replaces six input bits with four output bits based on a lookup table. The outputs of the s-boxes are then put through a “fixed permutation”, where each of the s-boxe’s output bits are spread across four different S-boxes in the next round. This produces a 32-bit result which is used as input for the next round. In the last round, the output of the last stage is not used as input, but is the resulting ciphertext.

DES is pretty easy to break with an exhaustive key search due to the small key size, so there came the need for a stronger encryption method. That method became TripleDES. TripleDES has a key length of 168-bits, which is actually 3x56-bit keys. The first stage is to encrypt the data using DES and the first key. The second stage is to decrypt the output of the first stage using DES and the second key. The third and final step is to encrypt it again using DES and the third key. The resulting 64-bit output is the ciphertext.

RC4 is an encryption algorithm that uses a shared key. RC4 uses a keystream which is a pseudo-random stream of bits. This is combined with the plaintext using bitwise XOR. To create the keystream, the cipher takes a permutation of all 256 possible bytes, then a KSA (key scheduling algorithm) is used to initialize the permutation with a variable length key that is typically between 40 and 256 bits. The stream of bits is then generated by a pseudo-random generation algorithm and XORed with the plaintext. The KSA works by setting the key length of the algorithm to the number of bytes in the key. The key can range from 1-256. The key length is usually between 5 and 16 and will corespoond to a key length of between 40 and 128 bits. The permutation of possible keys is then initialized with the identity permutation of the key. The pseudo-random generation works by choosing an output byte using a mathematical function and outputting it to be XORed with the plaintext. The algorithm will continue to output bytes from the keystream for as many iterations as required by the algorithm to completely encrypt the plaintext.

RC4-drop[n] was created to solve a problem in RC4 where there would be a certain number of bytes at the beginning of the keystream that were always the same after each encryption. This was fixed by stripping the first n bytes from the keystream. The rest of the algorithm is mostly the same as RC4.

Rabbit is similar to RC4 as it uses a bitstream generator that utilizes a 128-bit key and a 64-bit IV. The plaintext is encrypted in blocks of 128 bits. The encryption is performed by combining the bitstream with the plaintext with an XOR operation to create the ciphertext. It uses a slightly different way to get the random bitstream byte, but the process is essentially the same as RC4. It encrypts a block of 128 bits per iteration and rotates inner states between iterations. It uses a mixing function that entirely uses arithmetical operations that are available on modern processors so it is a very fast algorithm.

Out of all the encryption algorithms that this program uses, I would most likely only use AES and RC4-drop[n]. Rabbit is not a good choice since it is so similar to RC4 but has a smaller key size. Rc4-drop[n] is basically just a better version of it. I would not use DES because of the very small key length and its vulnerability to brute force attacks makes it a very bad choice. TripleDES is harder to crack than DES, but is still vulnerable to exhaustive key searches. TripleDES is also inferior to AES seeing as AES was created to replace DES and TripleDES. AES bettered the DES algorithm by being a symmetric and iterative block cipher. This makes it VERY strong against attempts to crack it. RC4-drop[n] is also a good choice because it gets rid of the vulnerability of RC4 where the first few bytes were always the same between encryptions. Since it uses a keystream to encrypt the plaintext, it is very secure and the encryption time is fast. This makes it ideal for SSL on websites and is the main reason that it is so widely used.